[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/851,592	05/09/2001	Bhashyam Ramesh	9491	2588

26890 7590 02/07/2006

JAMES M. STOVER
NCR CORPORATION
1700 SOUTH PATTERSON BLVD, WHQ4
DAYTON, OH 45479

EXAMINER	
CAO, DIEM K	
ART UNIT	**PAPER NUMBER**
2194	

DATE MAILED: 02/07/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/851,592	RAMESH ET AL.	
	Examiner	Art Unit	
	Diem K. Cao	2194	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 December 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *2-6,9,12-26 and 37-42* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *2-6,9,12-26 and 37-42* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

WILLIAM THOMSON
SUPERVISORY PATENT EXAMINER

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 8/27/01	6) ☐ Other: _____.

DETAILED ACTION

1. Claims 2-6, 9, 12-26, 37-42 are pending.

2. In view of the Appeal Brief filed on 12/1/2005, PROSECUTION IS HEREBY REOPENED. A new ground of rejection is set forth below.

To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) request reinstatement of the appeal.

If reinstatement of the appeal is requested, such request must be accompanied by a supplemental appeal brief, but no new amendments, affidavits (37 CFR 1.130, 1.131 or 1.132) or other evidence are permitted. See 37 CFR 1.193(b)(2).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 2-6, 9, 12-26 and 39-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kleinman (U.S. 6,128,640) in view of Cutler et al. (U.S. 5,598,562) further in view of IBM

(Event Control Block Wait and Post Semantics for Distributed Computing Environment / Posix Threads).

5. As to claim 9, Kleinman teaches a Unix operating system (UNIX, Posix; col. 1, lines 17-22 and col. 3, lines 50-54), a plurality of execution entities (threads; col. 3, lines 65-67), the plurality of execution entities including a first execution entity (inherent from multiple threads), an event control module adapted to create event objects representing respective events (The alternative API would consist of constructing a Thread_Exit_Event object, Timer_Expiration_Event; col. 7, lines 7-10, All event objects representing any type of event; col. 4, lines 46-47), the first execution entity to wait on plural events (The application can add … to be waited for in a container Alert; col. 7, lines 39-41 and Each of M threads … different events; col. 4, lines 15-20, and col. 8, lines 45-48), a data structure (non-null collection of Alert objects, each of which corresponds to an event; col. 4, lines 29-30) associated with the first execution entity (a thread blocks on a single Alert object of the container type; col. 4, lines 31-32 and A thread creates a container Alert object; col. 5, lines 26-27), the data structure containing information of the plural events that the first execution entity is waiting on (It has a non-null collection of Alert objects, each of which corresponds to an event; col. 4, lines 29-35), and a controller (processor) adapted to awaken the first execution entity by signaling the first execution entity in response to one or more event state changes of the states of the plural events (When a thread blocks on a single Alert object … these events occur; col. 4, lines 31-35, 50-53 and Because the thread is waiting … unblocking the thread; col. 8, lines 45-48).

6. However, Kleinman does not teach event having a state, the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events, and awaken the first execution entity according to the logical relationship specified by the indicator, each event object contains an indication of the state of the event, wherein the indication has a first state to indicate that the event has been signaled and a second state to indicate that the event has not been signaled, each event object has a type indication to indicate whether the event object state is to be automatically reset to the second state from the first state once the event has been signaled or to be manually reset to the second state from the first state by an explicit action.

7. Cutler teaches event having a state (Unsignalled or Signalled; col. 27, lines 6), the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events (The Executive Wait Multiple routine has two modes of operation which can be specified by the user; col. 28, lines 22-29), and awaken the first execution entity according to the logical relationship specified by the indicator (suspends the issuing program… become signaled; col. 28, lines 26-29 and col. 26, lines 50-55), each event object contains an indication of the state of the event (Wait Status field; col. 27, lines 55-60), wherein the indication has a first state to indicate that the event has been signaled and a second state to indicate that the event has not been signaled (Unsignalled or Signalled; col. 27, lines 3-8 and Wait Status field; col. 27, lines 55-60). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman and Cutler because it provides "waitable objects", which are objects used to synchronize the operation of

one or more processes with one another or with specified events, Cutler also teaches routines for generating new types of waitable objects without modifying the operating system's kernel (col. 2, lines 39-48).

8. IBM teaches each event object could be automatically reset to the second state from the first state once the event has been signaled (page 1, first paragraph), IBM also teaches modification to the Event Control Block to provide manually reset to the second state from the first state by an explicit action (page 2, second paragraph). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman, Cutler and IBM because it would improve the system of Kleinman with a simple, easy to understand, wait and post mechanism wherein the ECB can be created and destroyed, and attached to and detached from, and one created and/or attached to, the ECB can be posted or waited on (page 1).

9. As to claim 2, Kleinman teaches the event control module is adapted to define a collection/list for a first one of the event objects, the list having plural entries corresponding to plural execution entities waiting on the event represented by the first event object (col. 4, lines 29-45; because each event object has a corresponding Contained Alert object, and each Contained Alert object has a list of Container Alert objects, and each Container Alert object is associated with a thread that created it; col. 5, lines 26-27). Kleinman further teaches the AlertCollection class supports array-like functions (col. 6, lines 40-42). Although Kleinman does not teach a queue, it is well known in the art to implementing the collection/list as an array. It

would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman and well-known technique to implement a queue because queue provides a method to store data that can be retrieved or access at any time by many ways.

10. As to claim 3, Kleinman as modified teaches the event control module is adapted to further create second objects (Container Alert objects; col. 4, lines 27-30), wherein each entry of the queue comprises a link to a corresponding second object (Contained Alert object has a list of Container Alert objects; col. 4, lines 42-45), each execution entity to sleep on an associated second object to wait on the event represented by the first event object (a thread blocks on a single Alert object; col. 4, lines 31-32; lines 50-53 and the thread is waiting on the container Alert collection; col. 8, lines 45-48).

11. As to claim 4, Kleinman teaches each second object is defined by a condition variable (pthread_cond_t cond; col. 9, class Alert, line 15).

12. As to claim 5, Kleinman does not teach the controller signals each thread by signaling the condition variable. Kleinman teaches the controller signals each thread by signaling the second object (All event objects ... by one of these containers; col. 4, lines 46-56). Cutler teaches the controller signals each thread by signaling the condition variable (Kernel synchronization primitives ... A timer is changed to signaled; col. 27, lines 3-30). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman and Cutler because it provides a method for synchronization to help coordinate access

to resources or data without having to modify the wait service routines and the scheduler in the operating system's kernel.

13. As to claim 6, Kleinman teaches each second object is defined by a condition variable and a mutex (pthread_mutex_t mutex, pthread_cond_t cond; col. 9, class Alert, lines 15-16).

14. As to claim 12, Kleinman teaches collections/lists associated with corresponding event objects representing the events the first execution entity is waiting on, each collection/list containing an entry corresponding to the first execution entity (col. 4, lines 29-45; because each event object has a corresponding Contained Alert object, and each Contained Alert object has a list of Container Alert objects, and each Container Alert object is associated with a thread that created it; col. 5, lines 26-27 and col. 8, lines 45-48). Kleinman further teaches the AlertCollection class supports array-like functions (col. 6, lines 40-42). Although Kleinman does not teach a queue, it is well known in the art to implementing the collection/list as an array. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman and well-known technique to implement a queue because queue provides a method to store data that can be retrieved or access at any time by many ways.

15. As to claim 13, Kleinman teaches the event control module is adapted to define a barrier object (Container Alert objects; col. 4, lines 27-30), the first execution entity to sleep on the barrier object to wait on the plural events (the thread is waiting on the container Alert collection rather than any single contained Alert; col. 8, lines 45-48), the queue of each event object

containing a link to the barrier object (Contained Alert object has a list of Container Alert objects; col. 4, lines 42-45).

16. As to claim 14, Kleinman teaches the barrier object is defined at least by a condition variable (pthread_cond_t cond; col. 9, class Alert, line 15).

17. As to claim 15, Kleinman teaches the barrier object is defined at least by a condition variable and a mutex (pthread_mutex_t mutex, pthread_cond_t cond; col. 9, class Alert, lines 15-16).

18. As to claim 16, Kleinman teaches the event control module comprises a library (the alternative API; col. 7, lines 7-9 and Appendix A-C; col. 9-12).

19. As to claim 17, Kleinman teaches the execution entities comprise threads (threads; col. 3, lines 65-67).

20. As to claim 18, Kleinman teaches plural processes (process, subprocess; col. 3, line 65 – col. 4, line 2), each process associated with one or more threads (the threads that compose the process; col. 3, line 66), the event control module to create a local event to synchronize threads within a process and to create a global event to synchronize threads of different processes (exit of a subprocess, the exit of another thread, the completion of asynchronous file I/O … Posix conditional; col. 4, lines 1-6).

21. As to claim 19, Kleinman teaches the global event comprises named event (Timer_Expiration_Event; col. 7, lines 60-67).

22. As to claim 20, Kleinman does not teach a plurality of nodes, each node comprising one or more of the plurality of execution entities. IBM teaches distributed computing environment (title), it would have been obvious IBM also teaches a plurality of nodes, each node comprising one or more of the plurality of execution entities.

23. As to claim 39, Kleinman does not teach the indicator is settable to a first value to specify a logical AND relationship between the plural events, and in response to the first value of the indicator, the controller to awaken the first execution entity in response to all of the plural events waited on by the first execution entity being signaled. Cutler teaches there is a mode which can be specified by the user to indicate the logical AND relationship between the events, and awaken the execution entity in response to all of the plural events waited on by the execution entity being signaled (The Executive Wait Multiple ... become signaled; col. 28, lines 24-29).

24. As to claim 40, Kleinman does not teach the indicator is settable to a second value to specify a logical OR relationship between the plural events, and in response to the second value of the indicator, the controller to awaken the first execution entity in response to any of the plural events waited on by the first execution entity being signaled. Cutler teaches there is a mode which can be specified by the user to indicate the logical AND relationship between the events,

and awaken the execution entity in response to any of the plural events waited on by the execution entity being signaled (The Executive Wait Multiple ... become signaled; col. 28, lines 24-29).

25. As to claim 21, Kleinman teaches generate event objects in a Unix operating system environment representing events used for synchronizing execution entities in the system (All event objects representing any type of event; col. 4, lines 46-47 and abstract and UNIX, Posix; col. 1, lines 17-22 and col. 3, lines 50-54), provide a queue containing entries associated with a first event object (col. 4, lines 29-45; because each event object has a corresponding Contained Alert object, and each Contained Alert object has a list of Container Alert objects, and each Container Alert object is associated with a thread that created it; col. 5, lines 26-27), each entry associated with a corresponding execution entity (each Contained Alert object has a list of Container Alert objects, and each Container Alert object is associated with a thread that created it; col. 5, lines 26-27), the plural entries of the queue enabling plural execution entities to wait on the first event object (a thread blocks on a single Alert object of the container type; col. 4, lines 31-32).

26. However, Kleinman does not teach each event object having a state to indicate if the corresponding event has been signaled, selectively set a type variable to one of a fist value and a second value, the first value indicating that the first event object is of an auto-reset type, and the second value indicating that the first event object is of a manual reset type, and in response to the state of the first event object indicating the corresponding event has been signaled, automatically

clear the state of the first event object to an un-signaled state and awaken only one of the plural execution entities waiting on the first execution object in response to the type variable being set to the first value, and not clear the state of the first event object until manually cleared and awaken all threads waiting on the first event object in response to the type variable being set to the second value. Although Kleinman does not teach a queue, it is well known in the art to implementing the collection/list as an array. Cutler teaches each event object having a state to indicate if the corresponding event has been signaled (Unsignalled or Signalled; col. 27, line 6), and awaken one or more execution entities when the corresponding event has been signaled (col. 28, lines 15-29).). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman and Cutler because it provides "waitable objects", which are objects used to synchronize the operation of one or more processes with one another or with specified events, Cutler also teaches routines for generating new types of waitable objects without modifying the operating system's kernel (col. 2, lines 39-48).

27. IBM teaches each event object could be automatically reset to the second state from the first state once the event has been signaled (page 1, first paragraph), IBM also teaches modification to the Event Control Block to provide manually reset to the second state from the first state by an explicit action (page 2, second paragraph). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman, Cutler and IBM because it would improve the system of Kleinman with a simple, easy to understand, wait and post mechanism wherein the ECB can be created and destroyed, and

attached to and detached from, and one created and/or attached to, the ECB can be posted or waited on (page 1).

28. As to claim 22, Kleinman teaches the instruction when executed cause the system to further create barrier objects (An application can create a container Alert object … via the own routine; col. 6, lines 25-28 and Alert (), Alert (const Alert &); col. 9, class Alert), each execution entity waiting on a corresponding barrier object to wait on an event (the thread is waiting on the container Alert collection rather than any single contained Alert; col. 8, lines 45-48 and a container Alert object … to an event; col. 4, lines 28-30).

29. As to claim 23, Kleinman teaches the instructions when executed cause the system to create barrier objects by defining each barrier object based on a condition variable according to the Unix operating system (An application can create a container Alert object … via the own routine; col. 6, lines 25-28 and Alert (), Alert (const Alert &); col. 9, class Alert and pthread_cond_t cond; col. 9, class Alert, lines 15-16).

30. As to claim 24, Kleinman teaches the instructions when executed cause the system to create barrier objects by defining each barrier object based on a condition variable and mutex according to the Unix operating system (An application can create a container Alert object … via the own routine; col. 6, lines 25-28 and Alert (), Alert (const Alert &); col. 9, class Alert and pthread_mutex_t mutex, pthread_cond_t cond; col. 9, class Alert, lines 15-16).

31. As to claim 25, Kleinman as modified teaches the queue of the first event object contains entries pointing to the barrier objects of the plural execution entities waiting on the first event object (Contained Alert object has a list of Container Alert objects; col. 4, lines 42-45, a thread blocks on a single Alert object; col. 4, lines 31-32; lines 50-53 and the thread is waiting on the container Alert collection; col. 8, lines 45-48).

32. As to claim 26, Kleinman as modified teaches provide a routine associated with each event object, the routine of the first event object to traverse the queue of the first event object and to signal the barrier objects pointed to by the entries in the queue of the first event object (void *Thread_Exit_Event::wrapper (void *vp); col. 11, line 63 – col. 12, lines 13).

33. As to claim 41, Kleinman teaches provide a data structure containing information of the plural events waited upon by the first execution entity (non-null collection of Alert objects, each of which corresponds to an event; col. 4, lines 29-30), awaken the first execution entity in response to states of the plural events waited upon by the first execution entity (When a thread blocks on a single Alert object ... these events occur; col. 4, lines 31-35, 50-53 and Because the thread is waiting ... unblocking the thread; col. 8, lines 45-48). However, Kleinman does not teach the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events waited on by the first execution entity, and awaken the first execution entity according to the logical relationship specified by the indicator. Cutler teaches event having a state (Unsignalled or Signalled; col. 27, lines 6), the data structure further containing an indicator settable to one of plural values to specify respective

plural logical relationships between the plural events (The Executive Wait Multiple routine has two modes of operation which can be specified by the user; col. 28, lines 22-29), and awaken the first execution entity according to the logical relationship specified by the indicator (suspends the issuing program... become signaled; col. 28, lines 26-29 and col. 26, lines 50-55).

34. As to claim 42, see rejection of claim 39 above.

35. Claims 37 and 38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kleinman (U.S. 6,128,640) in view of Sun (Introduction to Library Functions).

36. As to claim 37, Kleinman teaches a Unix operating system (UNIX, Posix; col. 1, lines 17-22 and col. 3, lines 50-54), a plurality of execution entities (threads; col. 3, lines 65-67), a storage module containing an event class (a data storage medium 130; col. 3, lines 44-46 and sample event class, Thread_Exit_Event; col. 11-12, Appendix C), and a processor (processor; col. 3, lines 55-64) adapted to execute the event class to provide an event-based synchronization mechanism comprising one or more events on which the plural execution entities are able to sleep (col. 4, lines 15-53 and col. 7, lines 8-10, 38-41 and col. 8, lines 45-48).

37. However, Kleinman does not teach an event library. Sun teaches an event library (3SYSEVENT; page 6). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Kleinman and Sun because it would apply the well known technique in the art to the new invention.

38. As to claim 38, Kleinman teaches plural processes (process, subprocess; col. 3, line 65 – col. 4, line 2), each process associated with one or more threads (the threads that compose the process; col. 3, line 66), the event control module to create a local event to synchronize threads within a process and to create a global event to synchronize threads of different processes (exit of a subprocess, the exit of another thread, the completion of asynchronous file I/O ... Posix conditional; col. 4, lines 1-6).

Response to Arguments

39. Applicant's arguments with respect to claims 9, 21 and 37 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Diem K. Cao whose telephone number is (571) 272-3760. The examiner can normally be reached on Monday - Friday, 5:30AM - 2:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, William Thomson can be reached on (571) 272-3718. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Any response to this action should be mailed to:
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist at 571-272-2100.

Diem Cao

[illegible]
WILLIAM THOMSON
SUPERVISORY PATENT EXAMINER